What's in an Agreement? An Analysis and an Extension of WS-Agreement

Aiello, Marco; Frankova, Ganna; Malfatti, Daniela

Published in:
EPRINTS-BOOK-TITLE

What's in an Agreement? An Analysis and an Extension of WS-Agreement

Marco Aiello[1], Ganna Frankova[1], and Daniela Malfatti[2]

[1] Dept. of Information and Communication Technologies,
University of Trento, Via Sommarive, 14, 38100 Trento, Italy
{marco.aiello, ganna.frankova}@unitn.it
[2] Corso di Laurea in Informatica
University of Trento, Via Sommarive, 14, 38100 Trento, Italy
daniela.malfatti@studenti.unitn.it

Abstract. Non-functional properties of services and service compositions are of paramount importance for the success of web services. The negotiation of non-functional properties between web service provider and consumer can be agreed a priori by specifying an agreement. WS-Agreement is a recently proposed and emerging protocol for the specification of agreements in the context of web services. Though, WS-Agreement only specifies the XML syntax and the intended meaning of each tag, which naturally leads to posing the question of "What's in an Agreement?" We answer this question by providing a formal definition of an agreement and analyzing the possible evolution of agreements and their terms. From our analysis we identify ways in which to make an agreement more robust and long lived by proposing two extensions to the specification and supporting environment.

1 Introduction

Web Services (WS) are a set of technologies that allow the construction of massively distributed and loosely coupled applications. One of the most thought provoking issues in web services is that of automatically composing individual operations of services in order to build complex added-value services. The research on composition is well under way, but most of the focus is on functional properties of the composition, that is, how does one automatically compose? How does one enrich the services with semantic self-describing information? How does one discover the available services to use for the composition? If, on the one hand, this is crucial, on the other one, it is not enough. Non-functional properties of the composition are also of paramount importance in defining the usability and success of a composed service. Think for instance of desiring a service that performs a biological computation composing the services offered by a number of web service enabled machines. If the user knows that the composition is correct with respect to his goal, he will be satisfied with the answer he receives, but if the answer takes 3 years to be delivered to the user, the correctness is of little use. Therefore, the quality of a composed service is very important when interacting with an asynchronous system built out of independent components.

B. Benatallah, F. Casati, and P. Traverso (Eds.): ICSOC 2005, LNCS 3826, pp. 424–436, 2005.

With the term Quality of Service (QoS) we refer to the non-functional properties of an individual service, or a composition of services. The term is widely used in the field of networking. Usually it refers to the properties of availability and performance. In the field of web services, the term has a wider meaning. Any non-functional property which affects the definition and execution of a web service falls into the category of QoS, most notably, accessibility, integrity, reliability, regulatory, and security [15]. Dealing with QoS requires the study of a number of problems. One, the design of quality aware systems. Two, the provision of quality of service information at the level of the individual service. Three, ensuring that a promised quality of service is actually provided during execution. In [2], we addressed the first issue by using the Tropos design methodology, and the second one by resorting to WS-Policy to describe QoS properties. In this paper, we consider the second and third issues; in particular, we show how to provide a framework to negotiate the provision of a service according to a predefined QoS, and how to handle changes during the interactions of web services, and how to prevent the QoS conditions failure.

WS-Agreement is an XML based language and protocol designed for advertising the capabilities of providers and creating agreements based on initial offers, and for monitoring agreement compliance at run-time. The motivations for the design of WS-Agreement stem out of QoS concerns, especially in the context of load balancing heavy loads on a grid of web service enabled hosts [10]. However, the definition of the protocol is totally general and allows for the negotiation of QoS in any web service enabled distributed system. If, on the one hand, the proposal of WS-Agreement is a step forward for obtaining web service based systems with QoS guarantees, on the other hand, the protocol proposal is preliminary. The current specification [3] defines XML syntax for the language and protocol, and it gives a vague textual overview of the intended semantics, without defining a set of formal mathematical rules. Furthermore, a reference architecture is proposed to show how WS-Agreement are to be handled, [13]. Nevertheless, a formal analysis of what an agreement is still missing.

In this paper, we address the question *What's in an Agreement?* In particular, we provide a formal analysis of WS-Agreement by resorting to finite state automata, we provide a set of formal rules that tie together agreement terms and the life-cycle of an agreement. From the analysis, some shortcomings of the protocol become evident. Most notably, there is no checking of how close a term to being violated and, even more, breaking one single term of the agreement results in terminating the whole agreement, while a more graceful degradation is desirable. Therefore, we propose an extension of the protocol for which we provide appropriate semantics, that allows for providing warning before the violation of an agreement and eventually the renegotiation of running agreements by tolerating the break of a term.

Web service QoS issues are gaining attention and have been addressed in a number of recent works. Some approaches are based on the extension of the Web Service Description Language (WSDL) to define not only functional, but also non-functional properties of the service, e.g., [11]. The main idea of the

approach is simple: provide syntax to define terms which refer to non-functional properties of operations. The problem with this kind of approach is that the QoS definition is tied to the individual operation, rather than with the service as a whole; furthermore, there is no run-time support. Once a quality is defined, it can not be changed at execution time.

In [18], the authors propose to define WS QoS by using XML schemata that both service consumers and service providers apply to define the agreed QoS parameters. The approach allows for the dynamic selection of WS depending on various QoS requirements. On the negative side, the life-cycle of agreements is not taken into account, and it is not possible to define an expiration for a negotiation. The feasibility of using constraint programming to improve the automation of web services procurement is shown in [16]. A semantic web approach, in which services are searched on the basis of the quality of semantically tagged service attributes is presented in [17]. A predictive QoS model for workflows involving QoS properties is proposed in [6]. In [9], the authors propose a model and architecture to let the consumer rate the qualities of a service. In addition, the industry has proposed a number of standards to address the issue of QoS: IBM Web Service Level Agreement (WSLA) and HP's Web Service Management Language (WSML) are examples of languages used to describe quality metrics of services, [12]. A recent proposal is the specification of a new WS protocol, called Web Services Agreement Specification [3]. In [7], it is presented the Agreement-Based Open Grid Service Management (OGSI-A) model. Its aim is to integrate Grid technologies with Web Service mechanisms and to manage dynamically negotiable applications and services, using WS-Agreement. The WS-Agreement protocol proposal is supported by the definition of a managing architecture: CREMONA–An Architecture and Library for Creation and Monitoring of WS-Agreement [13]. The Web Services Agreement Specification defines the interaction between a service provider and a consumer, and a protocol for creating an agreement using agreement templates. The above approaches show that frameworks for QoS definition and management are essential to the success of the web service technology, but there are a number of shortcomings that still need to be addressed. First, no one has worked out a formal definition of what the semantics of a QoS negotiation should be. Second, the frameworks should be more flexible at execution time because actual qualities of services may change over time during execution.

The remainder of the paper is organized as follows. In Section 2, we present the WS-Agreement protocol defined in [3]. In Section 3, we propose a formal definition of an agreement and of its life-cycle. Section 4 is devoted to the presentation of an extension of WS-Agreement with the goal of improving the duration and tolerance of an agreement in execution. Preliminary experimental results are in Section 5. Concluding remarks are summarized in Section 6.

2 WS-Agreement

In order to be successful, web service providers have to offer and meet guarantees related to the services they develop. Taking into account that a guarantee depends on actual resource usage, the service consumer must request state-dependent guarantees from the service provider. Additionally, the guarantees on service quality must be monitored and service consumers must be notified in case of failure of meeting the guarantees. An agreement between a service consumer and a service provider specifies the associated guarantees. The agreement can be formally specified using the WS-Agreement Specification [3].

A WS-Agreement is an XML-based document containing descriptions of the functional and non-functional properties of a service oriented application. It consists of two main components that are the agreement Context and the agreement Terms. The agreement Context includes the description of the parties involved in the agreement process, and various metadata about the agreement. One of the most relevant components is the duration of the agreement, that is, the time interval during which the agreement is valid.

Functional and non-functional requirements are specified in the Terms section that is divided into Service Description Terms (SDTs) and Guarantee Terms. The first provides information to define the services functionalities that will be delivered under the agreement. An agreement may contain any number of SDTs. An agreement can refer to multiple components of functionalities within one service, and can refer to several services. Guarantee Terms define an assurance on service quality associated with the service described by the Service Description Terms. An agreement may contain zero or more Guarantee Terms.

In [8] a definition for guarantee terms in WS-Agreement is specified and a mechanisms for defining guarantees is provided. An agreement creation process starts when an agreement initiator sends an agreement template to the consumer. The structure of the template is the same as that of an agreement, but an agreement template may also contain a Creation Constraint section, i.e., a section with constraints on possible values of terms for creating an agreement. In [4] enabling of customizations of terms and attributes for the agreement creation is proposed. After the consumer fills in the template, he sends it to the initiator as an offer. The initiator decides to accept or reject the offer depending on the availability of resource, the service cost, and other requirements monitored by the service provider. The reply of the initiator is a confirmation or a rejection.

An agreement life-cycle includes the creation, termination and monitoring of agreement states. Figure 1 shows a representation of the life-cycle. When an agreement is created, it does not imply that it is monitored. It remains in an `not_observed` state until a service starts its execution. The semantics of the

Fig. 1. The life-cycle of a WS-Agreement

states is as follows: `not_observed`: the agreement is created and is in execution, but no service involved in the agreement is running; `observed`: at least one service of the agreement is running; and `finished`: the agreement has terminated either successfully or not.

3 What's in an Agreement?

The WS-Agreement specification provides XML syntax and a textual explanation of what the various XML tags mean and how they should be interpreted. Thank to the syntax, it is possible to prepare machine readable agreements, but a formal notion of agreement is missing. In this section, we formalize the notion of agreement by defining its main components.

Definition 1 (Term). *A* term t *is a couple* (s, g) *with* $s \in S$ *and* $g \in G$*, where* S *is a set of* n *services and* G *is a set of* m *guarantees.* $T \subseteq S \times G$ *is the set of the terms* t*.*

In words, a term involves the relationship between a service s and a guarantee g, not simply a specific tag of the agreement structure. If the service s appears in the list of services, which the guarantee g is applied to, it means that the couple (s, g) is a term. The number of terms varies between 0 and $n \cdot m$, where 0 means that there is no association between services and guarantees, and $n \cdot m$ indicates the case where each guarantee is associated with all services.

Definition 2 (Agreement). *An* agreement A *is a tuple* $\langle S, G, T \rangle$*, where* S *is a set of* n *services,* G *is a set of* m *guarantees, and* T *is the set of the terms* t*.*

In the following analysis, it is more convenient to consider the agreement as a set of *Terms* rather than a set of related services and guarantees. From the definition of WS-Agreement, we say that an agreement can be in one and only one of three states: `not_observed`, `observed` and `finished`.

Definition 3 (External State). *The* external state A_{es} *of an agreement* A *is an element of the set* {`not_observed`, `observed` *or* `finished`}*.*

We call the above state external, as it is the observable one. We also define an internal state of an agreement, which captures the state of the individual terms.

Definition 4 (Internal State). *The* internal state A_{is} *of an agreement* A *is a sequence of terms' states* $ts_1, \ldots, ts_p$ *of maximum size* $n \cdot m$*, where* $ts_i = (ss_j, gs_k)$ *represents the state of* g_k *guarantee with respect to the state of the* s_j *service. Service and guarantee states range over the following sets, respectively:*

- $ss_j \in$ {`not_ready`, `ready`, `running`, `finished`}*, and*
- $gs_k \in$ {`not_determined`, `fulfilled`, `violated`}*.*

From the definition of *Term*, we see that services and guarantees are related and we can define the internal state of an agreement, but it is necessary to

	terms are in state	state of the agreement	transitions
(A)	(1)	`not_observed`	(B)
(B)	(1)(2)	`not_observed`	(C) (E)
(C)	(1)(2)(3)	`observed`	(D)(E)(F)(G)
(D)	(1)(2)(3)(5)	`observed`	(F)(G)
(E)	(1)(2)(4)	`observed`	(F)(H)
(F)	(1)(2)(3)(4)(5)	`observed`	(H)
(G)	(5)	`finished`	
(H)	(1)(2)(3)(4)(5)(6)	`finished`	

Fig. 2. Transition table for the relation between internal and external states

distinguish between terms that have the same service and terms that have the same guarantee.

Proceeding in our goal of answering the question of what is in an agreement, we define the relationship between the internal and external state of an agreement A. First, we note that not all state combinations make sense. For instance, it has no meaning to say that a guarantee is `violated`, when a service is in a `not_ready` state. The only admissible combinations are the following ones.

(1) (`not_ready`, `not_determined`) (2) (`ready`, `not_determined`)
(3) (`running`, `fulfilled`) (4) (`running`, `violated`)
(5) (`finished`, `fulfilled`) (6) (`finished`, `violated`)

In theory, there are 63 possible combinations of states in which terms can be. That is, $\sum_{i=1}^{6} \binom{6}{i}$ all terms could be in state (1), or in state (2),... or in state (6); there could be terms in states (1) and (2), (1) and (3), and so on. But again, considering the definition of WS-Agreement in [3], one concludes that not all 63 combinations make sense. Furthermore, it is possible to extract the possible evolutions of these aggregated internal states.

When an agreement is created its external state is `not_observed`, while all services are `not_ready` and all guarantees are `not_determined`, i.e., state (1). In the next stage some services will be `ready` while others will still be `not_ready`, i.e., there will be terms in state (1) and (2). In this case, the external state is also `not_observed`. Proceeding in this analysis, one can conclude that there are 8 situations in which terms can be. We summarize these in the table in Figure 2. In the table, we also present the relation between the internal states and the external states, and the set of transitions to go from one set of states to another. The latter transitions are best viewed as an automaton (which is illustrated in [1]).

4 Extension of WS-Agreement

From the semantics and formal analysis presented in Section 3, inspecting the automaton provided, we note that if the agreement arrives into the states (E)

or (F) there is a non recoverable failure, and consequently an agreement termination. Even if one single term is violated, the whole agreement is terminated. Furthermore, when an agreement is running there is no consideration on *how* the guarantee terms are fulfilled. Our goal is to provide an extension of WS-Agreement and of its semantics in order to make agreements more long-lived, and robust to individual term violations. In [14] we provide appropriate XML syntax to implement the proposed extension, while an example of using a subset on a concrete case study (DeltaDator Spa, Trento) of the proposed extension can be found in [1].

We propose two extensions to WS-Agreement. The first is used to **(i) anticipate violations**, while the second is devoted to the **(ii) run-time renegotiation**. (i) WS-Agreement considers guarantees of a running service as fulfilled or violated. Nothing is said about how the guarantee is fulfilled. Is the guarantee close or far to being violated? Is there a trend bringing the guarantee close to its violation? We propose to introduce a new state for the agreement in which a warning has been issued due to the fact that one or more guarantees are likely to be violated in the near future. By detecting possible violations, one may intervene by modifying the run-time conditions or might renegotiate the guarantees which are close to being violated. (ii) The WS-Agreement specification does not contemplate the possibility of changing an agreement at run-time. If a guarantee is not fulfilled because of resource overload or faults in assigning availability to consumers, the agreement must terminate. For maintaining the service and related supplied guarantees, it is necessary to create another agreement and negotiate the QoS again. This approach wastes resources and computational time, and increases network traffic. The goal of negotiation terms is to have the chance to modify the agreement applying the negotiation terms rather than respecting the original agreement. Applying the negotiation terms means that the services included in the agreement will be performed according to the new guarantees.

4.1 Life-Cycle and Semantics for the Extended Agreement

To obtain the desired extensions, we expand the set of states in which an agreement and a guarantee term can be and thus update the transition system. More precisely, the definition of an agreement does not change with respect to Definition 2, the difference lies in the fact that the set of terms T is now extended with special *negotiation terms*. These terms are defined as in Definition 1, but have a different role, i.e., they specify new conditions that enable modification of guarantees at run-time.

To account for the new type of terms, we need to extend the definition of external and internal state of an agreement. The external states of an extended agreement are enriched by the `warned` state, `checked` state, the `revisited` state, and the `denied` state. We say that an agreement can be in one of seven states. `not_observed`, `observed` and `finished` have the same meaning as in WS-Agreement, Figure 1. An Agreement is in state `checked` when the monitoring system is checking its services and guarantees. From the `checked` state the agreement can go to five different states: to `finished` if the agreement finishes

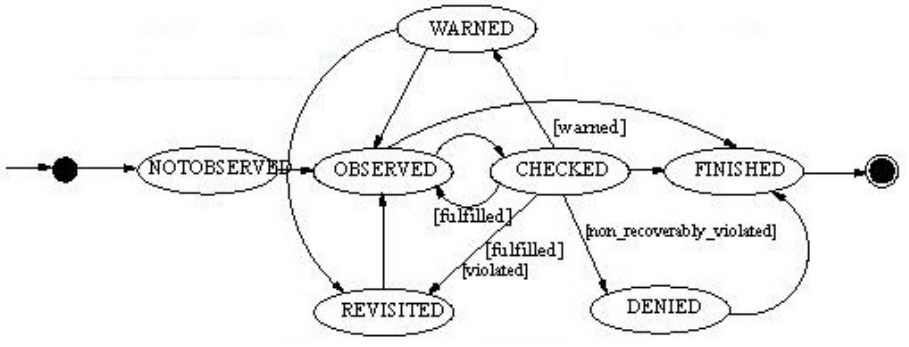

Fig. 3. The life-cylce of the WS-Agreement extension

its life-cycle; to `denied` if the agreement is violated and no *negotiation terms* can be applied, the agreement must terminate; to `warned` if the monitoring system has issued at least one warning for at least one term; back to `observed` if the agreement is fulfilled; to `revisited` if the agreement is fulfilled or violated and a *negotiation term* can be applied.

to `finished` if the agreement finishes its life-cycle;

Definition 5 (Extended External State). *The* extended agreement external state A_{xes} *of an agreement A is an element of the set* {`not_observed`, `observed`, `warned`, `checked`, `revisited`, `denied` *or* `finished`}.

The transitions between states are illustrated by the automaton in Figure 3, which is an extension of the one presented in Figure 1. The automaton represents the new evolution of an agreement where a guarantee can be modified during the processing of a service or a warning can be raised. When a guarantee is violated we have two situations: the first presents a recoverable violation which implies the chance to apply a negotiation term and so the agreement is in a revisited state, the second presents a non recoverable violation which implies that there is no suitable negotiation term for the current violated guarantee and so the agreement must terminate. Otherwise, if a warning is raised, this can be ignored or the agreement can go in a renegotiation state by ending in the revisited state. Also, when a guarantee is fulfilled, it is possible to change the current agreement configuration, applying a negotiation term that changes the QoS.

The internal state definition for the extended agreement is similar to the internal state definition stated before, but a new state for the services is added and two for the guarantees. A new state is `stopped` and is needed to define a state of a service where its associated guarantee is unrecoverable violated and the service must terminate or the guarantee can be revisited. It is an intermediate state. A guarantee can also be `warned` if it is close to being violated in a given time instant. Other state for a guarantee is the `non_recoverable_violated` state in which a guarantee is violated and it has no related negotiation terms for the current violation.

Definition 6 (Extended Internal State). *The* extended internal state A_{xis} *of an agreement* A *is a sequence of terms' states* $ts_1, \ldots, ts_p$ *of maximum size* $n \cdot m$*, where* $ts_i = (ss_j, gs_k)$ *represents the state of* g_k *guarantee with respect to the* s_j *service. Service and guarantee states range over the following sets, respectively:*

- $ss_j \in$ {`not_ready`, `ready`, `running`, `stopped`, `finished`}, *and*
- $gs_k \in$ {`not_determined`, `fulfilled`, `warned`, `violated`, `non_recoverably_violated`}.

As for Definition 4, one notes that not all the state combinations make sense. The only possible ones are the combinations itemized in Section 3 plus the following four:

(7) (`stopped`, `fulfilled`)
(8) (`stopped`, `violated`)
(9) (`stopped`, `non_recoverably_violated`)
(10) (`running`, `warned`)

The state combinations (7), (8) and (9) determine the states when a service is stopped because a guarantee is violated or is being modified. In state (7) a guarantee is fulfilled and we try to improve it applying a positive negotiation term. In (8) and (9) a guarantee is currently violated. In (8) the service is stopped and the guarantee is violated but it is possible to apply a negotiation term and to preserve the agreement again. In (9), instead, the guarantee is irrecoverably violated and the agreement must terminate, there are not any suitable negotiation terms. State (10) represents the fact that a warning has been raised for a running service guarantee.

The relation between internal and external states of an extended agreement is an extension of the one presented in the table in Figure 2, and it is presented in Figure 4. The table respects the original agreement evolution and presents some new transitions.

	terms are in state	state of the agreement	transitions
(A)	(1)	`not_observed`	(B)
(B)	(1)(2)	`not_observed`	(C)
(C)	(1)(2)(3)	`observed`	(D)(E)(F)(G)
(D)	(1)(2)(3)(5)	`observed`	(F)(G)(I)
(E)	(1)(2)(4)	`checked`	(F)(H)(I))
(F)	(1)(2)(3)(4)(5)	`checked`	(H)(I))(J)(K)(L)
(G)	(5)	`finished`	
(H)	(1)(2)(3)(4)(5)(6)	`finished`	
(I)	(1)(2)(3)(4)(5)(7)	`observed`	(D)(E)(F)(G)
(J)	(1)(2)(3)(4)(5)(8)	`revisited`	(D)
(K)	(1)(2)(3)(4)(5)(9)	`denied`	(F)(H)
(L)	(1)(2)(3)(5)(7)(10)	`warned`	(C)(D)(H)(I)(J))

Fig. 4. Extension of the transition table for the relation between internal and external states

4.2 Framework

The proposed extension to WS-Agreement must be handled by an appropriate framework that allows for monitoring and provides run-time renegotiation.

On the one hand, there must be rules specifying when and how to raise a warning for any given guarantee. These rules should be easy to compute to avoid overloading of the monitoring system and be fast to provide warnings. In addition they should provide good performance in detecting as many violations as possible generating the minimum number of false positives. A forecasting method which enjoys this characteristics is the linear least squares method [5]. The method of linear least squares requires a straight line to be fitted to a set of data points such that the sum of the squares of the vertical deviations from the points to the line is minimized. By analyzing such a parameter of the line as a slope ratio, it is possible to predict a change over time.

On the other hand, to allow for renegotiation of guarantee terms at run-time the parties involved in the agreement need to be able to decide whether a renegotation has been agreed upon. Before execution it must be possible to specify negotiation terms. This can be done by using appropriate templates in the spirit of the original work in [13].

5 Preliminary Experimental Results

We have conducted preliminary experimentation to show the feasibility of the warning strategy. We used synthetic data. We generated a sequence of 1100 elements considered as a service guarantee for a single operation over a continuous time interval (for instance the cost of a service which should be below the value 10). The data set and the complete results of the experiments are available at `http://www.dit.unitn.it/~frankova/ICSOC05_Exp/`. The points were generated by a function that returns a random number greater or equal to 6.00 and less or equal to 14.00, evenly distributed. We split the data set into two subsets. The first part of the data set was used to decide the size of the time window and of the threshold values to be used for prediction. The rest of the data was used for evaluating the system.

To evaluate the method we consider the following performance measures: *Precision* is the ratio of the number of true warnings (i.e., warnings thrown to notify violation points) to the number of total warnings (i.e., true warnings and false warnings). *Recall* is the ratio of the number of warned violations (i.e., violation points for which a warning is issued) to the number of total violation points. Total violation points include warned violations and missed violations.

The following table summarizes the results of the experimentation:

	Warnings		**Violations**	
	True	***False***	***Warned***	***Missed***
	303	11	156	13
Total	314		169	
Precision	**96.50%**			
Recall			**92.31%**	

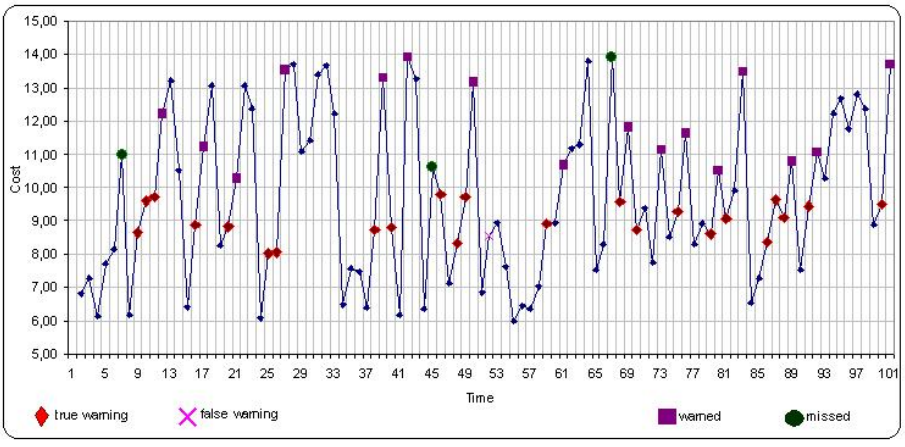

Fig. 5. Experimental results for 100 points

The number of true and false warnings is shown in the first column. The difference in the number of total warnings and violations is due to the fact that more than one warning in the same time window may refer to the same violation. The number of warned and missed violations is reported in the second column of the table. The total sum of warnings and violations is in the "Total" row. The last two rows present the precision and recall of the method.

The results of experimentation on the first 100 points of the data set is shown in Figure 5. In the figure, two types of warnings, true and false, are marked by diamonds and crosses, respectively. A warning is thrown if the cost and tangent of the cost curve are higher then the threshold (8 for cost and 0.1 for the tangent differences). Squares represent warned violation points, while circles indicate missed violation points.

The method shows good performance when the increase in cost is smooth (points 8, 9, and 10), a case that normally takes place during web services execution. If the change in values is abrupt then the method fails to generate warnings, e.g., points 43 (cost is 6.36) and 44 (cost is 10.63). It is difficult to find a violation point if the point is in the very beginning of the process, within or just after the first time window (point 7). The latter cases should be considered exceptional, in fact those occur only 13 times in the whole experiment.

In the experimentation using the method, more than 92% of violation points are warned in advance, and 96.5% of thrown warnings are true warnings. Using bigger time windows does not improve performances, see `http://www.dit.unitn.it/~frankova/ICSOC05_Exp/`.

6 Concluding Remarks

Describing and invoking an individual functionality of a web service is becoming more and more common practice. One of the next steps is moving from functional

properties of basic services to non-functional properties of composed services. The non-functional properties need to be specified by the services, but also to be negotiated among services.

WS-Agreement is a protocol that defines a syntax to specify a number of guarantee terms within an agreement. We looked into the protocol specification with the goal of providing a formalization of the notion of an agreement and proposing a formal representation for the internal and external states in which an agreement can be. From this analysis we discovered that an agreement can be made more long-lived and robust with respect to forecoming violations. We presented the details of the proposed extension in formal terms and provided some preliminary experimentation on synthetic data.

This work prods for more investigation of agreements and of their management. In the next future, we plan to dive into the details of a framework implementing the extended agreement version and then to experiment on real data coming from an actual case study.

Acknowledgments

Marco thanks Asit Dan and Heiko Ludwig for useful discussion on WS-Agreement while visiting IBM TJ Watson.

References

1. M. Aiello, G. Frankova, and D. Malfatti. What's in an agreement? A formal analysis and an extension of WS-Agreement. Technical Report DIT-05-039, DIT, University of Trento, 2005.
2. M. Aiello and P. Giorgini. Applying the Tropos methodology for analysing web services requirements and reasoning about Qualities of Services. *CEPIS Upgrade - The European journal of the informatics professional*, 5(4), 2004.
3. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). Technical report, Grid Resource allocation Agreement Protocol (GRAAP) WG, 2004.
4. A. Andrieux, A. Dan, K. Keahey, H. Ludwig, and J. Rofrano. Negotiability constraints in WS-Agreement. Technical report, Grid Resource Allocation Agreement Protocol (GRAAP) Working Group Meetings, 2004.
5. Rudolf K. Bock. *The data analysis : briefbook*. Springer: Berlin [etc.], 1998.
6. J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Journal of Web Semantics*, 2004. To appear.
7. K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based Grid Service Management (OGSI-Agreement). Technical report, Global Grid Forum, GRAAP-WG Author Contribution, 2003.
8. A. Dan, K. Keahey, H. Ludwig, and J. Rofrano. Guarantee Terms in WS-Agreement. Technical report, Grid Resource Allocation Agreement Protocol (GRAAP) Working Group Meetings, 2004.
9. V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. A quality of service management framework based on user expectations. In *Service-Oriented Computing (ICSOC)*, pages 104–114. LNCS 2910, Springer, 2003.

10. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer*, 35(6), 2002.
11. D. Gouscos, M. Kalikakis, and P. Georgiadis. An approach to modeling web service QoS and provision price. In *1st Web Services Quality Workshop (WQW2003) at WISE*, 2003.
12. H. Ludwig. Web services QoS: External SLAs and internal policies or: How do we deliver what we promise? In *1st Web Services Quality Workshop (WQW2003) at WISE*, 2003.
13. H. Ludwig, A. Dan, and R. Kearney. CREMONA: an architecture and library for creation and monitoring of ws-agreements. In M. Aiello, M. Aoyama, F. Curbera, and M. Papazoglou, editors, *ICSOC*, pages 65–74. ACM, 2004.
14. D. Malfatti. A framework for the monitoring of the QoS by extending WS-Agreement. Master's thesis, Corso di Laurea in Informatica, Università degli Studi di Trento, 2005. In Italian.
15. A. Mani and A. Nagarajan. Understanding quality of service for web services, 2002. `http://www-106.ibm.com/developerworks/library/ws-quality.html`.
16. O. Martn-Daz, A. Ruiz Corts, A. Durn, D. Benavides, and M. Toro. Automating the procurement of web services. In *Service-Oriented Computing (ICSOC)*, pages 91–103. LNCS 2910, Springer, 2003.
17. M. P. Singh and A. Soydan Bilgin. A DAML-based repository for QoS-aware semantic web service selection. In *IEEE International Conference on Web Services (ICWS 2004)*, 2004.
18. M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller. A concept for QoS integration in web services. In *1st Web Services Quality Workshop (WQW2003) at WISE*, 2003.